QSP RAMDISK

(C) DENNIS SKALA

DIST. BY MICROCOM SOFTWARE

COCO 3 OS-9 RAMDISK PACKAGE
(written by Dennis Skala)

DISKETTE TABLE OF CONTENTS:

```
dup                   duplication utility

LEV1 (directory)

    rdisk              Level I ramdisk service module
    ram.dr               "   " ramdisk device driver
    r0
128.dd             "   " ramdisk device descriptor - 128K machine
    r0
512.dd             "   "    "       "         "       - 512K machine

LEV2 (directory)

    rdisk              Level II ramdisk service module
    ram.dr               "   "  ramdisk device driver
    r0
64.dd              "   "  ramdisk device descriptor - 64K default size
    r0
128.dd             "   "      "       "         "       - 128K    "      "
    r0
192.dd             "   "      "       "         "       - 192K    "      "
    r0
256.dd             "   "      "       "         "       - 256K    "      "
```

* N.B. Although Level I & Level II files have the same filenames, these are different files. Do not mix. The dup utility works on both Level I and Level II.

INTRODUCTION

A ramdisk is a portion of memory which is reserved as a "pretend disk drive." The main advantage of a ramdisk is speed. A ramdisk is especially useful with programs which use a lot of disk I/O. The Microware C compiler is an example of such a program - one which uses many temporary disk files. Many word processors and database managers also fall into this category. Using a ramdisk as the default device greatly speeds up the operation of such programs. The principal disadvantage to a ramdisk is its volatility. All data is lost when the power goes off. System crashes can also be catastrophic (although this software does offer a recover function, results cannot be guaranteed). Ramdisk files ahould therefore be backed up frequently.

The modular, open structure of OS-9 is particularly suited to a ramdisk. On boot-up, the entire OS-9 operating system is essentially built from scratch. OS-9 can easily be customized to suit individual tastes and individual

destination device will be left free and available. If the source device is larger than the destination device, there are two possibilities. If the largest used LSN on the source device is smaller than the size of the destination device, duplication will proceed as outlined above. If not, a warning will be issued, and the user may still proceed, knowing that some information will be lost.

The dup utility permits rapid transfer of an entire ramdisk to a floppy disk and vice-versa, regardless of the format of either one. For instance, you may keep a copy of your CMDS directory on a separate floppy disk, and rapidly transfer it to a ramdisk. Then chx /r0/cmds will allow quick loading of commands which are not in memory. Extra memory may be assigned to dup by means of the usual #xx entry on the command line. Such extra memory is used in the copy buffer to speed up duplication. Dup itself uses a $1500 byte data area.

Another use for the dup utility is to quickly transfer the contents of an entire floppy disk to another floppy disk which may be formatted differently. For instance, Level II OS-9 permits the reading of various disk formats automatically. If you're using double-sided 40 track disks in your system, you may make a backup copy of a single-sided 35 track disk (such as that you would probably get commercial software on) to your disk format with the command

```
dup
```

Here the duplication would be from the disk in drive 0 to the disk in drive 1. Dup does not include a single drive option.

MISCELLANEOUS

You may want to include rdisk and/or dup in your startup file. If you include these in any command file, remember that they do require user interaction, and the standard input should be redirected to your keyboard, unless you're absolutely certain in all cases what the desired response will be. In a Level II command file, use

```
rdisk </1
```

This will redirect rdisk to use the current standard input path for Level II. For a Level I command file, use </term.

To use dup in your startup file, say after a ramdisk is formatted, include the lines:

```
dup -v /d1 /r0
yy
```

In this case, dup will take its input (two "yes" answers) from the command file, specifically from the next line.

If you have external terminals on your system, it would probably be a good idea to remove public execute access on rdisk. Only the super-user should have permission to alter the system resources.

DPS
8/25/87
rev. 5/4/88

TROUBLESHOOTING

It is impossible to anticipate every problem which could occur, but here are some obvious things to try:

Won't boot - use ident on the os9boot file to make sure all modules are correct. If all else fails, try a differently ordered boot file. This sometimes fixes Level II boot problems.

Ramdisk won't install - make sure you haven't inadvertently interchanged Level I and Level II modules. Is rdisk in the current CMDS directory? Do an mdir command to make sure ram and r0 are present in memory.

Recovery problems - try increasing the start block value as described above. Increment by one or two and make a new boot disk. When OS-9 starts up, it uses some scratch memory temporarily. Depending on the exact makeup of your system, the $0F block may be getting clobbered during startup, even though it shows free at the first OS-9 prompt.

Memory lockout - deiniz of a formatted ramdisk will permanently lockout a large section of memory. If this happens, reset the machine and recover the ramdisk if desired.

THE DUP UTILITY

A duplication utility is included in this package which works identically under Level I or Level II. Its operation is similar to the backup utility supplied with OS-9, except that the source and destination devices need not be formatted identically. Both must be RBF devices, and both must have fewer than $800 sectors. Dup will work only on 40 track double-sided and smaller disks. It will not work on 80 track double-sided or larger disks. The format is

```
dup <-v> </source device> </destination device>
```

The "-v" option must appear first if used. It acts to prevent a verification pass after the duplication is complete. Such verification is merely a sector-by-sector read of the destination device, and is mainly intended to verify that the physical structure of a disk is intact (format and backup do the same thing). It is unnecessary if the destination device is a ramdisk.

The source device will default to /d0 and the destination device to /d1 if no device parameters are present on the command line. If only one device parameter is present, it is assumed to be the source, and the destination device will default to /d1.

Both devices must be formatted for dup to work. If the source device is smaller (has a lesser number of logical sectors) than the destination device, a sector to sector copy will be performed, and the excess sectors on the

hardware configurations. New hardware can easily be added to an OS-9 system - new software to control the hardware is simply appended to the system. Likewise, unneeded system software can easily be removed so that it doesn't waste memory. To use existing memory as a ramdisk, one simply needs to add a device driver and device descriptor to the system.

With the advent of the COCO 3, we have enough memory within the machine itself to consider using some of it as a ramdisk. In the case of a 128K machine, this isn't too much - under Level I, 64K is available; under Level II, only about 32K may be used in any sort of practical fashion. Using Level I (version 02.00.00 only) on a 512K COCO 3 leaves 448K of memory available for a ramdisk. With Level II, available memory on a 512K machine will depend on system resources and on what programs need be run. Under most conditions, 256K should be quite practical. If only small programs are to be run, a 320K ramdisk is easily useable.

Under Level I, all available extra memory over 64K might just as well be used for a ramdisk - OS-9 doesn't even know it's there, and otherwise it is just wasted. Under Level II, one must compromise between memory available for modules/data and that used for a ramdisk. In this case, it is advantageous to have user selectable ramdisk size and to allow user termination of the ramdisk. In either case, OS-9's hierarchical file structure makes multiple ramdisks somewhat redundant. If one desires a logical split between groups of files, two subdirectories will easily accomplish this. The OS-9 dsave utility can be used to copy a part of a ramdisk to permanent storage.

INSTALLATION

As with any new software, it is a good idea to immediately make a backup copy of the disk, and put the original away for safekeeping. Do all your work with the copy.

Level I:

It is best to put all frequently used device drivers and descriptors in the boot file. They use less memory this way, since the boot file is tightly packed. Alternatively, one may load drivers and descriptors as needed, and unlink them when not needed. If you choose to do the latter, it is more convenient and less wasteful of memory to merge the driver and descriptor into one file, e.g.

```
   chd /d1/LEV1
   merge ram.dr r0
512.dd > ramdisk
   attr /d1/LEV1/ramdisk e
```

The 'attr' command is necessary because a file created with a 'merge' command never has execution permission (regardless of what was merged). Hence it cannot be loaded into memory until this permission is explicitly given.

Then to use the ramdisk,

```
load /d1/LEV1/ramdisk;link r0  (use entire pathlist on first)
```

```
   /d1/LEV1/rdisk                  (use entire pathlist)
```

Or for a single drive system, do the following:

```
   load load merge attr link
      (insert the ramdisk software disk)
   chd /d0/LEV1
   merge ram.dr r0
512.dd > ramdisk
   attr /d0/LEV1/ramdisk e
   load /d0/LEV1/ramdisk;link r0  (use entire pathlist on first)
   /d0/LEV1/rdisk                 (use entire pathlist)
      (replace your system disk)
   unlink load merge attr link
   chd /d0
```

Of course the 'merge' and 'attr' commands are only needed the first time you do this - once the ramdisk file is created, you may simply load it. When you are done with the ramdisk and want to free up the memory that the driver and descriptor use,

```
   unlink ram r0
```

The easiest way to include the driver and descriptor in the bootfile is to copy the driver ram.dr and both r0
xxx.dd descriptors to the MODULES
directory on your working copy of the Boot/Config disk. Then run config as per instructions in your OS-9 manual. Be sure to only select one r0 descriptor. The driver will only support one ramdisk. Alternatively, you may use os9gen to add the ramdisk driver and one descriptor to your existing bootfile. After your new boot disk is complete, add the rdisk module to /D0/CMDS.

Having the driver and descriptor in the bootfile makes the ramdisk available for your use immediately upon boot-up without any explicit action on your part. This is much more convenient, and less wasteful of memory. The driver and descriptor use approximately 500 bytes of memory.

Note that the disk file named ram.dr contains a module named ram, and the disk files named r0
xxx.dd contain modules named r0. The suffixes on the
disk file names conform to the standard OS-9 designations for driver and descriptor filenames as used on the boot/config disk.

Level II:

Under Level II, it is even more important that drivers and descriptors be included in the bootfile, since loading them manually will consume at least 8K of memory, most of which is wasted. However for occasional use, they may be loaded and run after booting up. To do this, use a procedure similar to the above, i.e.

```
   chd /d1/LEV2
   merge ram.dr r0
256.dd rdisk >ramdisk
```

If you're running Basic09, you may want to 'dup' over your entire Basic09 disk to the ramdisk, chd /r0;chx /r0/cmds, and work entirely off the ramdisk. Loading in graphics from disk would also seem to be a natural application for a ramdisk.

***** WARNING *****

Do NOT deiniz /r0 while the ramdisk is formatted (Level II only). This will result in the loss of the ramdisk, and will also permanently lock out of the system those memory blocks which were in the ramdisk. If you want to recover the data area memory of the ramdisk driver in your system map, you may deiniz /r0, but do so only after terminating any existing ramdisk.

PATCHES

If you don't like the name r0, you may change this in the Level I software in the following places:

* in rdisk, at offset $4F.

* in r0, at offset $21 (hi bit set in last character).

If you do this, be sure to change both, and to use the OS-9 verify utility to update the CRC values of both modules. A maximum of two characters may be used for the device name. The OS-9 debug utility may be used to perform the patches. To set the hi bit, add $80 (decimal 128) to the ASCII code for the character.

In the level II code, the appropriate locations are:

* in rdisk, at offset $246.

* in r0, at offset $21 (hi bit set in last character).

Here, the Level II modpatch utility must be used. Again, be sure to use the "v" option in modpatch before exiting to update the CRC values. To set the hi bit, add $80 (decimal 128) to the ASCII code for the character.

Another value which it may be desirable to patch in some cases in the Level II code is the byte in the device driver which determines the lowest block number to be assigned to the ramdisk. As supplied, this byte contains the value $0E, one less than the lowest possible ramdisk block, $0F (decimal 15). The location is:

* in ram, at offset $140   (Level II only).

The default ramdisk size is obtained from the device descriptor. Although four different descriptors are included, you may want to patch your own. This one byte value is the default number of 8K blocks to be allocated to the ramdisk, located

* in r0, at offset $18.

```
rdisk
```

This will install a ramdisk with the volume name 'Ramdisk'. The device descriptor used will be r0, and the ramdisk size will come from r0.

```
rdisk t
```

This will terminate any existing ramdisk.

```
rdisk 20 'Important Data' /r1
```

This will install and format a ramdisk with the volume name 'Important Data', using a device descriptor named r1 (you would have had to provide this on your own). 20 blocks maximum (160K) will be allocated for the ramdisk.

```
rdisk r
```

This will recover the ramdisk /r0 after a reset or termination.

USING THE RAMDISK

OK, now that you have a ramdisk in your system, what do you do with it? Well, anything you can do with a floppy disk, you can do with the ramdisk, only faster and more quietly. As far as OS-9 is concerned, the ramdisk looks exactly like another floppy disk drive. To start with, try a directory. Use rdisk as described above to install/format a ramdisk, then type

```
dir /r0
```

You should get a blank directory, since the ramdisk is newly formatted. Also try

```
free /r0
```

This should show the size of the ramdisk, as well as its volume name. Now try copying files to the ramdisk and deleting them. If you have a large textfile, copy it to the ramdisk and list it out. Or try the /d0/startup file.

```
copy /d0/startup /r0/startup
list /r0/startup
```

Want to copy your entire commands directory to the ramdisk and execute system commands from the ramdisk? Try the following:

```
chd /d0/CMDS; dsave -s18 /d0 ! (chd /r0)
```

When this finishes, chx /r0/cmds and try a few system commands. They will all be loaded from the ramdisk if not in memory already. An alternative procedure would be to use the dup utility described below to more rapidly transfer an entire disk to the ramdisk.

```
attr /d1/LEV2/ramdisk e
```

Again, the 'attr' command is necessary to explicitly set the execution permission for the file named ramdisk, and allow its loading into memory. Here we have included the ramdisk service module in the merge.

Then to use the ramdisk,

```
load /d1/LEV2/ramdisk;link r0  (use entire pathlist on first)
rdisk
```

Or for a single drive system, do the following:

```
load load merge attr link
   (insert the ramdisk software disk)
chd /d0/LEV2
merge ram.dr r0
256.dd rdisk > ramdisk
attr /d0/LEV2/ramdisk e
load /d0/LEV2/ramdisk;link r0  (use entire pathlist on first)
rdisk
   (replace your system disk)
unlink load merge attr link
chd /d0
```

Of course the 'merge' and 'attr' commands are only needed the first time you do this - once the ramdisk file is created, you may simply load it. When you are done with the ramdisk and want to free up the 8K block of memory that the driver, descriptor, and rdisk use, do the following:

```
rdisk t
deiniz r0
unlink ram r0 rdisk   (may have to be repeated, and
                       may end with error message)
```

It is preferable to include the driver and descriptor in the bootfile. To do this, copy the driver ram.dr and all the r0 xxx.dd descriptors to the MODULES directory on your working copy of the Boot/Config disk. Then run config as per instructions in your OS-9 manual. Be sure to select only one r0 descriptor. The driver will only support one ramdisk. Choose the descriptor which represents the default ramdisk size of your choice (you may always override the default size in setting up the ramdisk). Alternatively, you may use os9gen to add the ramdisk driver and one descriptor to your existing bootfile. After your new boot disk is complete, add the rdisk module to /D0/CMDS.

Having the driver and descriptor in the bootfile makes the ramdisk available for your use immediately upon boot-up without any explicit action on your part. This is much more convenient, and less wasteful of memory. The driver and descriptor use approximately 700 bytes of system memory.

Note that the disk file named ram.dr contains a module named ram, and the disk files named r0 xxx.dd contain modules named r0. The suffixes on the

disk file names conform to the standard OS-9 designations for driver and descriptor filenames as used on the boot/config disk.

USING RDISK

Level I:

The rdisk module is a ramdisk service module. It installs the ramdisk in formatted form. The ramdisk size is taken from the device descriptor. The Level I rdisk requires no parameters. There is no need to terminate the ramdisk, since the memory which it uses is not otherwise recognized at all by Level I OS-9. To clear the ramdisk, simply re-install it again with rdisk. To use rdisk (assuming that it is in your default execution directory), type

```
rdisk
```

The computer will respond with:

```
OK to format ramdisk? (Y/N)
```

If you respond with "Y", a new ramdisk will be installed, destroying any existing ramdisk. A "N" response will abort the procedure, leaving any existing ramdisk intact. Since the Level I ramdisk uses memory outside that recognized by OS-9, it will survive a warm reset. So long as the power is not turned off, ramdisk contents will be intact after a reset.

Although the OS-9 format command will work with the Level I code, it is unnecessary, since rdisk installs the ramdisk in formatted form, ready to use, in less than a second.

Level II:

The Level II rdisk ramdisk service module is somewhat more complicated. It may use up to 4 parameters. The format is

```
rdisk <'name'> </device> <size in blocks> <action>
```

All parameters are optional, and their order is unimportant. Only spaces may serve as separation between parameters. 'name' is the volume name, and must be contained within single quote marks. Default is 'Ramdisk'. /device is the ramdisk device name - default is /r0. Size in blocks is the ramdisk size in decimal. Each memory block in the COCO 3 is 8K, so using a value of 10 would result in an 80K ramdisk. If no size parameter is given, the default ramdisk size is taken from the device descriptor. A maximum of 40 out of the 64 blocks in a 512K machine may be allocated for the ramdisk, regardless of the default size value. If you request more blocks than are available in the current system block map, rdisk will allocate as many blocks as it can. The total number of blocks allocated is reported at the completion of rdisk.

"Action" (no quotes in the command line), if used, must be either the character "r" or "t". If neither "r" or "t" appears on the command line, a new ramdisk will be installed and formatted by default.

The "t" action parameter is used to terminate an existing ramdisk and return its memory to the system memory pool. When the "t" option is used, all other parameters are unimportant, and are ignored. To guard against inadvertant termination, rdisk will ask for verification.

The "r" action parameter will cause restoration to be attempted, so long as a current ramdisk does not exist. For restoration, only the /device parameter has any meaning, and any other parameters on the command line are ignored. Restoration is never guaranteed, but if done immediately after a termination or immediately upon reset, will work in the majority of cases. For instance, if a running program crashes the system in an orderly fashion (that is to say does not destroy any memory outside of its own assigned address space), you may recover the ramdisk by doing the following:

* Press reset.
* Reboot the computer in exactly the same way it was previously booted - use the same boot disk.
* When the OS-9: prompt first appears, type

```
rdisk r
```

If all goes well, you should be informed that the ramdisk is recovered. A warning also appears to remind you to check file structure of any critical files. List text files, or use the ident utility on executable files to do this.

You may even be able to recover after running other modules, but the odds are not as good. The reason is that the system uses memory blocks of its own choice for module and data areas. If it uses one of the blocks which was previously in your ramdisk, it will write over any information which was contained there. If it happens to overwrite the root directory of the ramdisk, no recovery is possible. If the directory structure is intact, the the ramdisk is apparently recovered, but some of the information therein may be destroyed. If any of the blocks which used to contain the ramdisk is unavailable when recovery is attempted, recovery is aborted with a message to that effect.

For all practical purposes, only a few blocks may be allocated for a ramdisk in a 128K machine. Four would seem to be about the maximum practical, but this could vary, depending on what modules are in the bootfile. If too many blocks are requested on a 128K machine, there will simply not be any memory left over to do anything. If this happens, terminate the ramdisk and try a smaller number (rdisk should run, since it was running when the ramdisk was installed).

Again, the OS-9 format command is unnecessary, since rdisk installs the ramdisk in formatted form, ready to use, in less than a second. The OS-9 format command will not work with the level II ramdisk.

Some examples of the use of rdisk: